Data Management Challenges in Production Machine Learning

Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, Martin Zinkevich

Google Research

{npolyzotis, sudipr, swhang, martinz}@google.com

ABSTRACT

The tutorial discusses data-management issues that arise in the context of machine learning pipelines deployed in production. Informed by our own experience with such large-scale pipelines, we focus on issues related to understanding, validating, cleaning, and enriching training data. The goal of the tutorial is to bring forth these issues, draw connections to prior work in the database literature, and outline the open research questions that are not addressed by prior art.

CCS Concepts

•**Information systems** → **Data management systems;**
•**Computing methodologies** → **Machine learning;**

1. INTRODUCTION

It is hard to overemphasize the importance of machine learning in modern computing. More and more organizations adopt machine learning as a tool to glean knowledge from data and tackle a diverse set of computationally hard tasks, ranging from machine perception, to text understanding, health care, genomics, and even the protection of endangered species [2].

In many cases, machine learning refers to the one-off application of a learning algorithm on a specific dataset. In these scenarios, the user of machine learning is typically a data scientist or analyst who wishes to test drive machine learning or uses it to mine information from the data. Our focus here is different and considers the deployment of machine learning in production. This involves setting up a pipeline that reliably ingests training datasets as input and generates a model as output, in most cases doing so continuously (so that new models are generated as fresh datasets arrive) and dealing gracefully with different types of failures. This scenario typically involves a team of engineers who spend a significant portion of their time on the less glamorous aspects of machine learning like maintaining and monitoring the machine learning pipelines [16]. The goal of this tutorial is to describe the data-management issues that arise in these machine learning pipelines, draw connections to existing works in the database literature, and outline the open problems that remain to be solved.

As an example, consider the fundamental problem of ensuring that the data is "valid". (We discuss a few notions of validity later.) Invalid data will result in bad models which in turn will affect downstream services in uncontrolled (and almost always negative) ways. The database community has long worried about data validity and has developed robust mechanisms to deal with this issue, e.g., by enforcing a schema over the data or devising ways to detect unclean data. However, as we argue in this tutorial, production machine learning pipelines do not always provide the scaffolding required by existing solutions. More generally the context of machine learning introduces new dimensions in these long-standing problems.

The target audience of the tutorial comprises database researchers and practitioners. The goal is to inform the audience about the class of problems that exist in the intersection of production machine learning pipelines and data management, and to motivate further research in this area. We believe that the database community is well positioned to tackle these data-related problems.

Note that the tutorial does not cover issues related to the performance of machine learning pipelines, when the latter are viewed as dataflows that can be optimized by applying techniques from database query processing. We acknowledge that this is a rich topic with numerous interesting and unsolved problems. However, our goal is to draw attention to a different class of problems that match naturally a core expertise of the database community: analyzing, modeling, enriching, validating, and debugging data.

In what follows, we describe at a high level the characteristics of a production machine learning pipeline and then review a few of the data-related problems that we cover in the tutorial.

2. PRODUCTION MACHINE LEARNING: OVERVIEW AND ASSUMPTIONS

Figure 1 shows a high-level schematic of a production machine learning pipeline. The input of the system comprises the training datasets that will be fed to the machine learning algorithm. The output is a machine-learned model that is then picked up by serving infrastructure and used in conjunction with serving data in order to generate predic-

SIGMOD'17 May 14-19, 2017, Chicago, IL, USA

ACM ISBN 978-1-4503-4197-4/17/05.
DOI: http://dx.doi.org/10.1145/3035918.3054782

tions. Typically a subset of the serving data, along with the model's predictions, is transformed and fed back into the system as new training data.

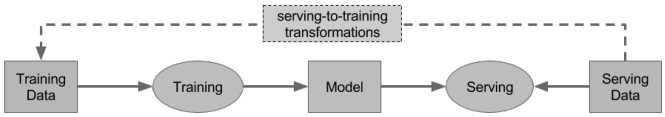

Figure 1: Simplified schematic of a production machine learning pipeline

We focus on the front part of the pipeline and specifically on issues related to managing the training data. Our experience has shown that these issues can affect critically the quality of the generated models and also require significant time investment to address when they occur. We assume that the training dataset can be modeled as a (large) collection of flat records, each record comprising a large number of features that take scalar values. This model is predominant in practice and is also adopted by several existing machine learning frameworks (including the open-sourced Tensorflow [1] library).

We assume a semi-continuous setup where training data arrives in batches, the machine learning algorithm runs on a sliding window of recent batches, and so a new batch of data triggers the generation of a new model. Many of the problems that we will describe below are also relevant in a pure streaming setup as well as for one-off executions on a single batch of data.

To simplify exposition, the schematic deliberately omits several modules that perform critical tasks, such as testing the generated model against holdout data, replacing a model that is already being served, or doing dark model launches, to name a few. There are interesting data-related problems to examine in these modules as well, but they are not part of this tutorial due to time constraints.

3. DATA ISSUES IN PRODUCTION MACHINE LEARNING

In this section, we discuss key challenges in managing data for production machine learning pipelines.

3.1 Understanding

Engineers setting up the machine learning pipeline for the first time spend a significant portion of their time analyzing their raw data. This process involves generating and visualizing salient features about the data (e.g., the range and statistical distribution of feature values, correlations between features in the training data, and distributions of positive and negative examples across different slices) and identifying any anomalies or outliers that exist in the data. Scaling this process to large training data (e.g., a few terabytes per day) can be a daunting task. Techniques developed for online analytical processing [9, 15, 14], data-driven recommendation of visualizations [17], and approximate query processing [8, 3] can be applied to develop tools that make it easier for users to understand their own data.

Another significant step for engineers is to figure out how to encode their data into features that are compatible with the trainer. For example, if a string feature of the raw data contains country identifiers (e.g., "US", "CN", "AU", and so on) then it can be converted to an integer feature using one-hot encoding. Automatically recommending and generating transformations from raw data to features based on characteristics of the data is an interesting and largely unexplored research area.

Context also plays an important role in understanding data. As stated in [16], it is crucial to clearly identify explicit and implicit data dependencies (and accordingly avoid needlessly introducing new dependencies) in order to develop a maintainable machine learning pipeline. Many of the techniques developed for data-provenance management are applicable for tracking some of these dependencies, and can therefore help us understand how data flows through these complex pipelines. At the same time, machine learning brings new twists to the problem, such as identifying the short-term and long-term impacts of removing a legacy feature from the pipeline, or tracking provenance when the data is generated and processed through a highly-heterogeneous infrastructure (which is typical in machine learning pipelines).

3.2 Validation

It is hard to overlook the fact that data validity affects crucially the quality of the generated model. The notion of validity has several facets, including: ensuring that training data have the expected features; these features have the expected values; features are correlated as expected; and, serving data does not deviate from training data.

Some of these issues can be solved through well-known mechanisms from database systems. For instance, the expected features and the characteristics of their values can be encoded with something akin to a *schema* for the training data. As another example, checking for correlations among features is very similar to checking for functional dependencies and inclusion dependencies in relational databases. However, the specific context of machine learning introduces a few wrinkles. For instance, the schema describing the constraints may be a property of the pipeline but may not be part of the process generating the data, which means that the data and the schema can evolve separately. (Constrast this with traditional DBMS technology where the data always conforms to the schema.) Moreover, machine learning introduces unique types of constraints that need to be checked, e.g., bounds on the drift in the statistical distribution of feature values in the training data, or use of an embedding for some input feature if and only if other features are normalized in a specific fashion. Furthermore, unlike a traditional DBMS, any schema over training data needs to be flexible enough to permit changes in the characteristics of training data as they reflect events in the real world (for example, a spike in queries about politics prior to elections). The schema definition language for training data should be expressive enough to allow (or disallow) such episodic changes in the training data as well as detect gradual drifts in the training data over a period of time.

The deviation between serving and training data (often referred to as training-serving skew) is a major source of problems in production machine learning pipelines. The underlying issue is that the data used to generate the model is different than the data used to probe the model, which (almost invariably) means that the generated predictions are incorrect. This difference can manifest in many ways: the two types of data may have different features, these features

may use different value encodings (e.g., using 0 and 1 for boolean values in the training data vs. "0" and "1" in the serving data), the value distribution for specific features is different, or there is *time travel* in the feature values. The latter arises when the serving data is post-processed before being used for training (see the "serving-to-training transformations" box in Figure 1) and this post-processing uses information that is available after serving (hence, the time travel) to alter the feature values. Several works in the data-management literature are relevant for skew detection, e.g., time-series analysis [5] can be used to detect distribution drifts, but the detection of time-traveling is a unique machine learning problem that may require different solutions.

There are two orthogonal factors that add more complexity to these validation tasks. The first is the scale of the training data, since it is typical for machine learning pipelines to train on billions of examples at a time each having hundreds of features. Dealing with this scale may require solutions that lift techniques from approximate query answering [8]. The second complicating factor is that validity checks may need to be done at the level of data slices, rather than in aggregate. An example scenario is a machine learning pipeline that pulls data from different sources but a data slice from one particular source is invalid. It is clearly useful to identify this slice and help the user localize the error. However, it is non-trivial to detect the error in the aggregated data from all data sources, determine the features on which to slice the data, and finaly identify a specific slice that explains the error.

It is clearly useful to be able to identify the slices containing errors, but the challenge is that the identity of slices may not be available and in aggregate the data can still look valid. This problem is related to the identification of interesting slices in OLAP analytics [15]. In the context of machine learning pipelines, the measures of interest should correlate with metrics that users actually care about and are likely to monitor, such as prediction error or other loss metrics.

It is worth mentioning here that the aforementioned validity errors can appear for different reasons, e.g., due to bugs in the code that collects or transforms the data, miscommunication between groups handling different parts of the pipeline, or due to production hiccups such as the deployment of binaries with inconsistent versions. In other words, while it is conceptually possible to avoid these errors before they affect the training data, a variety of reasons makes it necessary to have checks after the data gets ingested in the machine learning pipeline.

3.3 Cleaning

Once a validation error is detected, the next logical step is to clean the data. We break down this task in three sequential subtasks: understanding where the error occurred; understanding the impact of the error; and, fixing the error.

We illustrate the first subtask (understanding where the error occurred) through an example. Assume that the data is invalid because the value distribution of some specific feature(s) has changed significantly over time. A simple diagnosis of "feature X has different distribution" may not be very helpful–there is no context to help localize where the drift occurred. Contrast this with a diagnosis of "feature X has different distribution in training examples where Y takes values in [20, 40)". This localization can help an engineer understand whether this is indeed an error or a natural evolution of the data (in which case the right course of action is to update the validation checks). Doing this localization is equivalent to understanding which regions of the data are interesting or relevant for the detected anomaly, and so it may be possible to leverage previous works on guided OLAP exploration [14] (although the objective function needs to reflect the validation constraints related to machine learning).

The second subtask is to understand the impact of the error on model quality. For practical reasons, a team may be willing to continue operating the pipeline with invalid data if the impact on model quality is negligible. Quantifying this impact is non-trivial in the general case, given that the machine learning algorithm is in principle a black box function. Here, we may be able to leverage some characteristics of the function for specific classes of machine learning algorithms in order to derive the impact analytically, or run a controlled number of experiments to quantify the impact empirically. The latter is of course challenging due to the overhead of running these experiments, but we may be able to leverage previous works on DB-parameter optimization [6] in order to schedule these experiments effectively.

The third task is actually cleaning the data to fix the error. This cleaning can be done by addressing the root cause, e.g., fixing a bug in the code that generates the data. An alternative is to patch the data inside the machine learning pipeline, as a temporary fix until the underlying problem is permanently fixed. This approach is related to the rich body of work on database repairs for certain types of constraints [4]. A recent study [12] also looked at applying similar techniques for a specific class of machine learning algorithms. An open question is whether these techniques can be extended to cover the class of validation constraints discussed in Section 3.2 and also to a more general class of machine learning algorithms (including artificial neural networks).

3.4 Enrichment

Enrichment refers to the augmentation of the training and serving data with new features in order to improve the quality of the generated model. A common form of enrichment is to join in a new data source in order to augment the existing features with new signals. Another form is using the same signals with different transformations, e.g., using a new embedding for text data.

A key problem in this context is discovering which additional signals or transformations can enrich the data in a meaningful way. Having a catalog of sources and signals can provide a first step for discovery, and recent works have considered the problem of data cataloguing in different contexts [10, 11] and the discovery of relationships between sources. An equally important problem is helping the team understand the boost in model quality by enriching the data with a certain set of features. This information will help the team decide whether to invest resources in implementing the enrichment in production. (Typically, the difficulty lies in enriching the serving data, where latency requirements can be stringent.) A recent paper [13] has looked into this problem for the case of joining with new data sources and a specific class of algorithms, and it would be interesting to consider extensions to other cases (e.g., training with black-box learning algorithms that are hard to approximate, or using different transformations on existing signals).

Another wrinkle is that data sources may contain sensitive information (e.g., PII data) and hence may not be accessed until the team goes through an access review. However, going through a review and obtaining access to sensitive data can result in operational overheads (e.g., the team has to log accesses for auditing and enforce certain policies). Hence, it is interesting to ask whether the effect of enrichment can be approximated in a privacy-preserving fashion and without access to the sensitive data, in order to help the team decide whether to apply for access. One possibility is to leverage techniques from privacy-preserving learning [7], although the focus of previous works is more on learning a privacy-preserving model rather than approximating the effect of additional features on model quality.

4. REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[2] R. Abousleiman, G. Qu, and O. A. Rawashdeh. North atlantic right whale contact call detection. *CoRR*, abs/1304.7851, 2013.

[3] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*, pages 29–42, 2013.

[4] L. Bertossi. Consistent query answering in databases. *SIGMOD Rec.*, 35(2):68–76, June 2006.

[5] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, Aug. 2008.

[6] S. Duan, V. Thummala, and S. Babu. Tuning database configuration parameters with ituned. *PVLDB*, 2(1):1246–1257, 2009.

[7] C. Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, volume 4978, pages 1–19. Springer Verlag, April 2008.

[8] M. N. Garofalakis and P. B. Gibbons. Approximate query processing: Taming the terabytes. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, 2001.

[9] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[10] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods: Organizing google's datasets. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 795–806, New York, NY, USA, 2016. ACM.

[11] J. M. Hellerstein, V. Sreekanti, J. E. Gonzales, Sudhansku, Arora, A. Bhattacharyya, S. Das, A. Dey, M. Donsky, G. Fierro, S. Nag, K. Ramachandran, C. She, E. Sun, C. Steinbach, and V. Subramanian. Establishing common ground with data context. In *Proceedings of CIDR 2017*, 2017.

[12] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12):948–959, 2016.

[13] A. Kumar, J. F. Naughton, J. M. Patel, and X. Zhu. To join or not to join?: Thinking twice about joins before feature selection. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 19–34, 2016.

[14] S. Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 307–316, 2000.

[15] G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional OLAP data. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 531–540, 2001.

[16] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young. Machine learning: The high interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[17] M. Vartak, S. Rahman, S. Madden, A. G. Parameswaran, and N. Polyzotis. SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13):2182–2193, 2015.